Machine Learning for Neuroscience

06/07/2016

Mariya Toneva
mariya@cmu.edu

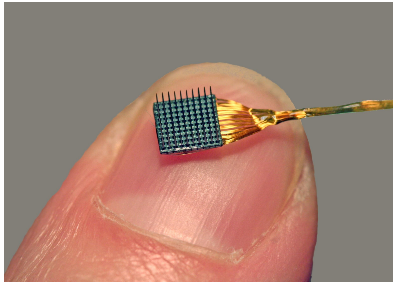
Some figures derived from slides by Tom Mitchell, Aarti Singh, Ziv Bar-Joseph, and Alona Fyshe

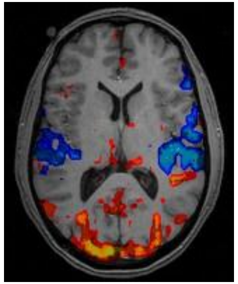
Goal: intuitive understanding of ML methods and how to use them

- we'll use scikit-learn: http://scikit-learn.org/stable/
- brief homeworks after each class, both critical thinking and using scikit-learn
- video tape each lecture and put videos on Youtube after each class

How can ML help neuroscientists?

- ❏ Deal with large number of sensors/recording sites

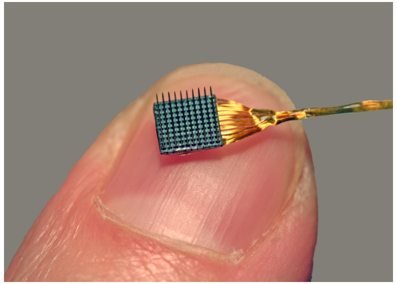


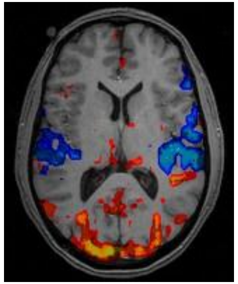


  - ❏ investigate high-dimensional representations

How can ML help neuroscientists?

- Deal with large number of sensors/recording sites

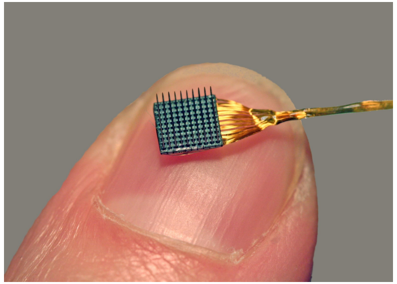


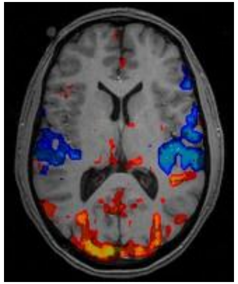


  - investigate high-dimensional representations
    - classification (what does this high-dimensional data represent?)

How can ML help neuroscientists?

- Deal with large number of sensors/recording sites

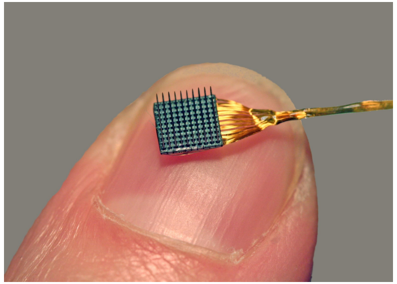


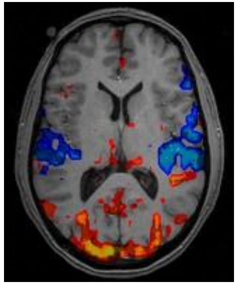


- investigate high-dimensional representations
  - classification (what does this high-dimensional data represent?)
  - regression (how does it represent it? can we predict a different representation?)

How can ML help neuroscientists?

- Deal with large number of sensors/recording sites

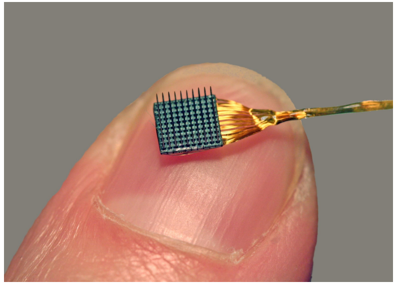


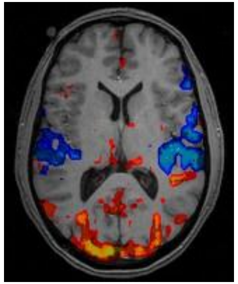


  - investigate high-dimensional representations
    - classification (what does this high-dimensional data represent?)
    - regression (how does it represent it? can we predict a different representation?)
    - model selection (what model would best describe this high dimensional data?)

How can ML help neuroscientists?

- Deal with large number of sensors/recording sites

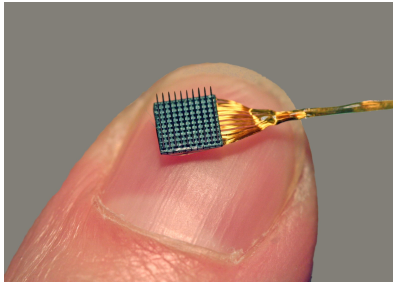


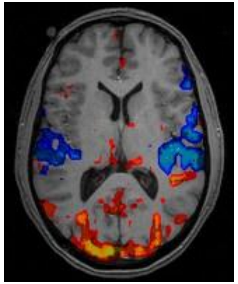


  - investigate high-dimensional representations
    - classification (what does this high-dimensional data represent?)
    - regression (how does it represent it? can we predict a different representation?)
    - model selection (what model would best describe this high dimensional data?)
  - uncover few underlying processes that interact in complex ways

How can ML help neuroscientists?

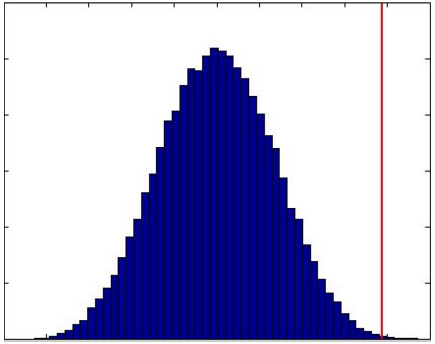
- Deal with large number of sensors/recording sites





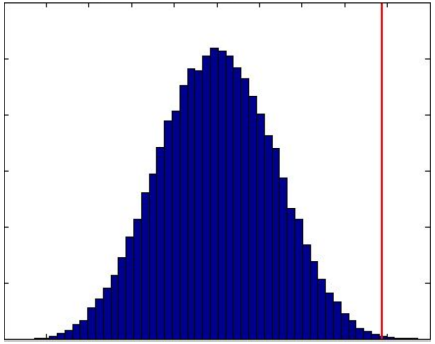
  - investigate high-dimensional representations
    - classification (what does this high-dimensional data represent?)
    - regression (how does it represent it? can we predict a different representation?)
    - model selection (what model would best describe this high dimensional data?)
  - uncover few underlying processes that interact in complex ways
    - dimensionality reduction techniques

How can ML help neuroscientists?



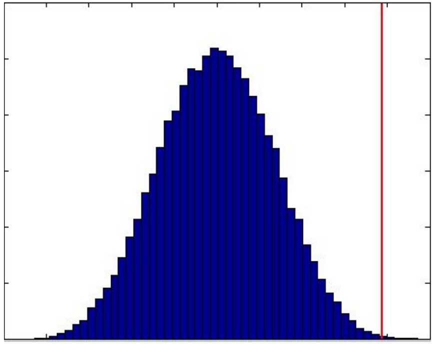
- Evaluate results

How can ML help neuroscientists?



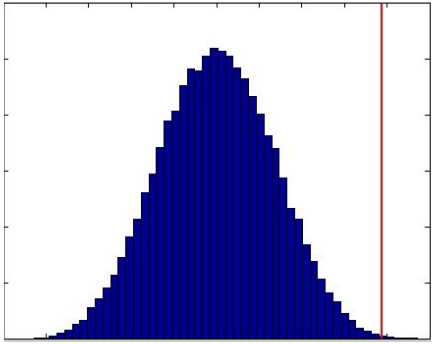
- Evaluate results
  - cross validation (how generalizable are our results?)

How can ML help neuroscientists?



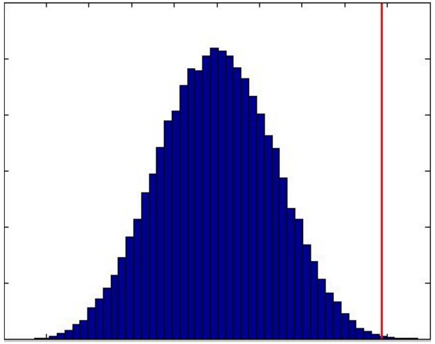
- Evaluate results
  - cross validation (how generalizable are our results?)
  - nearly assumption-free significance testing (are the results different from chance?)

How can ML help neuroscientists?

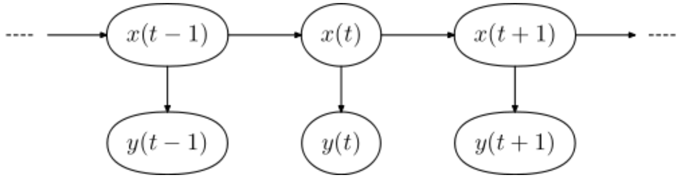


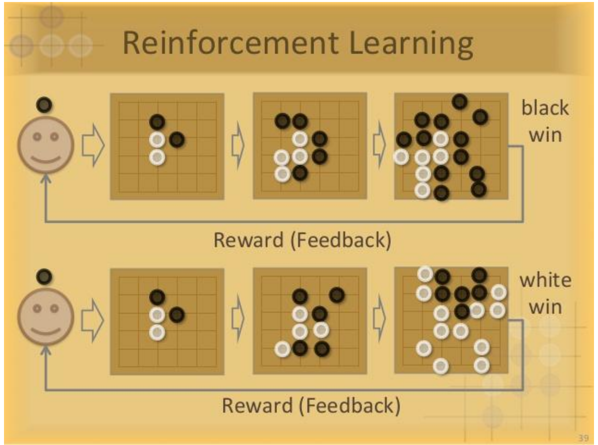
- ❑ Evaluate results
  - ❑ cross validation (how generalizable are our results?)
  - ❑ nearly assumption-free significance testing (are the results different from chance?)
- ❑ Complex data-driven hypotheses of brain processing

How can ML help neuroscientists?



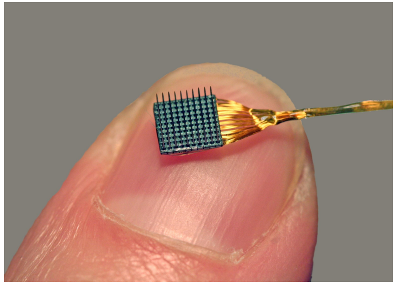
- Evaluate results
  - cross validation (how generalizable are our results?)
  - nearly assumption-free significance testing (are the results different from chance?)
- Complex data-driven hypotheses of brain processing
  - advanced topics: latent variable models, reinforcement learning, deep learning

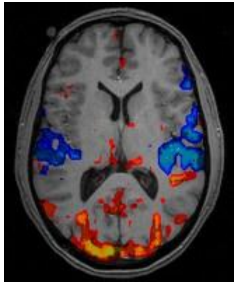




Today: classification and regression

- ❏ Deal with large number of sensors/recording sites





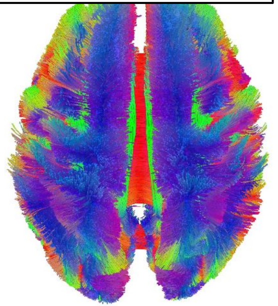
- ❏ investigate high-dimensional representations
  - ❏ **classification (what does this high-dimensional data represent?)**
  - ❏ **regression (how does it represent it? can we predict a different representation?)**
  - ❏ model selection (what model would best describe this high dimensional data?)
- ❏ uncover few underlying processes that interact in complex ways
  - ❏ dimensionality reduction techniques

Today: classification and regression

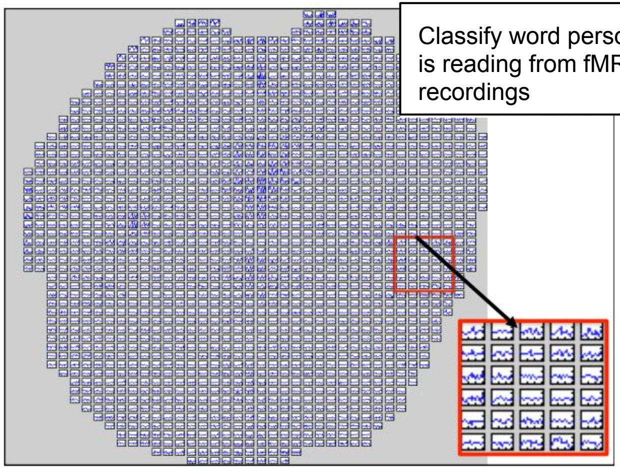
- Classification methods:
  - naive Bayes
  - support vector machine (SVM)
  - k-nearest-neighbors (kNN)
- Regression:
  - linear

Classification: example problems

Classify DTI fiber tracks into anatomical bundles



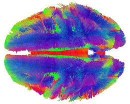
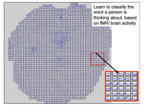
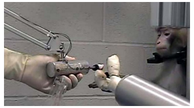
Classify word person is reading from fMRI recordings



Classify direction of desired movement from motor cortex recordings

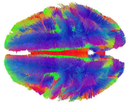
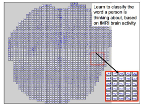
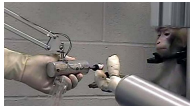
What is a classifier?

- ❑ Any algorithm that can assign a class label to each data instance

			
class labels			
data instance			

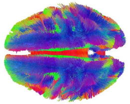
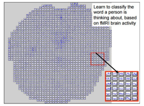
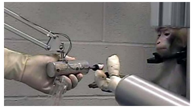
What is a classifier?

- Any algorithm that can assign a class label to each data instance

			
class labels	“cerebellum”, … “anterior_commisure”		
data instance			

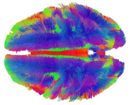
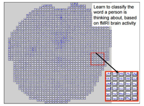
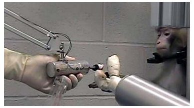
What is a classifier?

- Any algorithm that can assign a class label to each data instance

			
class labels	“cerebellum”, … “anterior_commisure”		
data instance	<(x_1,y_1,z_1), … (x_t,y_t,z_t)>		

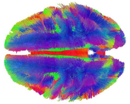
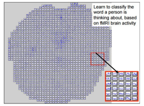
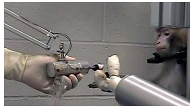
What is a classifier?

- Any algorithm that can assign a class label to each data instance

			
class labels	“cerebellum”, … “anterior_commisure”	“chair”, … “celery”	
data instance	<(x_1,y_1,z_1), … (x_t,y_t,z_t)>		

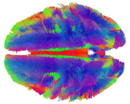
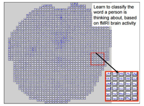
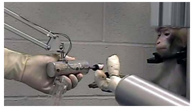
What is a classifier?

- Any algorithm that can assign a class label to each data instance

			
class labels	"cerebellum", … "anterior_commisure"	"chair", … "celery"	
data instance	$<(x_1,y_1,z_1), \dots (x_t,y_t,z_t)>$	$<v_1,v_2,\dots,v_n>$	

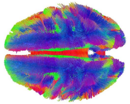
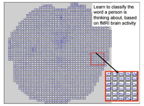
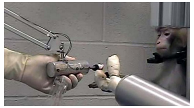
What is a classifier?

- Any algorithm that can assign a class label to each data instance

			
class labels	“cerebellum”, … “anterior_commisure”	“chair”, … “celery”	“ ↗ “, … “ ↖ “
data instance	<(x_1,y_1,z_1), … (x_t,y_t,z_t)>	<$v_1,v_2,...,v_n$>	

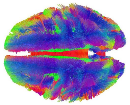
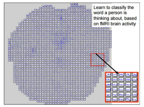
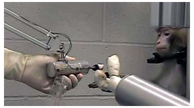
What is a classifier?

- Any algorithm that can assign a class label to each data instance

			
class labels	“cerebellum”, … “anterior_commisure”	“chair”, … “celery”	“ ↗ “, … “ ↖ “
data instance	$<(x_1,y_1,z_1), \dots (x_t,y_t,z_t)>$	$<v_1,v_2,\dots,v_n>$	$<n_1,n_2,\dots,n_n>$

What is a classifier?

- Any algorithm that can assign a class label to each data instance

			
class labels	“cerebellum”, … “anterior_commisure”	“chair”, … “celery”	“ ↗ “, … “ ↖ “
data instance	<(x_1,y_1,z_1), … (x_t,y_t,z_t)>	<$v_1,v_2,\ldots,v_n$>	<$n_1,n_2,\ldots,n_n$>

- Input: feature vector for each data instance
- Output: class label

Classifiers divide in roughly 3 types

- Generative
  - Build a generative statistical model
    - Naive Bayes
- Discriminative
  - Directly estimate a decision rule or boundary
    - SVM
- Instance based classifiers
  - Use observations directly without building a model
    - kNN

How do most classifiers work? 3 main steps!

Assume: Make an assumption about the data

- e.g. assume all values of voxels recorded during the presentation of the same word follow a Gaussian distribution

How do most classifiers work? 3 main steps!

Assume: Make an assumption about the data

- e.g. assume all values of voxels recorded during the presentation of the same word follow a Gaussian distribution

Train: Use data to estimate (learn) the parameters of this model

- e.g. estimate the mean and covariance of the Gaussian for each word

How do most classifiers work? 3 main steps!

Assume: Make an assumption about the data

- e.g. assume all values of voxels recorded during the presentation of the same word follow a Gaussian distribution

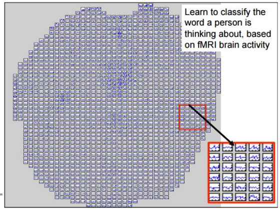
Train: Use data to estimate (learn) the parameters of this model

- e.g. estimate the mean and covariance of the Gaussian for each word

Test: Apply learned model to new data

- e.g. using the learned mean and covariance for each word class, calculate how likely it is for a new data instance to belong to each class

Problem with train step: too many parameters to estimate can lead to their inaccurate estimation

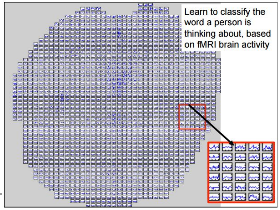


Let voxels have only 2 possible values: 0 and 1
Let us be interested in a region of interest with 30 voxels
Let the person only read 2 words: “chair” and “celery”

v_1	v_2	...	v_{29}	v_{30}	w
0	1	...	1	0	“chair”
0	1	...	1	1	“celery”
1	1	...	0	1	“celery”

- Parameter = P(w = “chair”$|v_1=0,v_2=1,...v_{29}=1,v_{30}=0$) =>

Problem with train step: too many parameters to estimate can lead to their inaccurate estimation



Let voxels have only 2 possible values: 0 and 1
Let us be interested in a region of interest with 30 voxels
Let the person only read 2 words: “chair” and “celery”

v_1	v_2	...	v_{29}	v_{30}	w
0	1	...	1	0	“chair”
0	1	...	1	1	“celery”
1	1	...	0	1	“celery”

- Parameter = P(w = “chair”|v_1=0,v_2=1,...v_{29}=1,v_{30}=0) => ~2^{30} parameters
- Need more instances than parameters to estimate correctly => lots of data!

Reduce number of parameters using Bayes rule

$$\mathbb{P}(w|v_1, v_2, \dots v_{29}, v_{30}) = \frac{\mathbb{P}(v_1, v_2, \dots v_{29}, v_{30}|w)\mathbb{P}(w)}{\mathbb{P}(v_1, v_2, \dots v_{29}, v_{30})}$$

- How many parameters for $P(v_1,v_2,...v_{29},v_{30}|w)$? $\sim 2^{30} x2$!

Reduce number of parameters using Bayes rule

$$\mathbb{P}(w|v_1, v_2, \ldots v_{29}, v_{30}) = \frac{\mathbb{P}(v_1, v_2, \ldots v_{29}, v_{30}|w)\mathbb{P}(w)}{\mathbb{P}(v_1, v_2, \ldots v_{29}, v_{30})}$$

- How many parameters for $P(v_1,v_2,...v_{29},v_{30}|w)$? $\sim 2^{30} x2$!
- Naive Bayes assumption: all features are independent given a class

$$\mathbb{P}(v_1, v_2, \ldots v_{29}, v_{30}|w) = \prod_{i=1}^{30} \mathbb{P}(v_i|w)$$

Reduce number of parameters using Bayes rule

$$\mathbb{P}(w|v_1, v_2, \ldots v_{29}, v_{30}) = \frac{\mathbb{P}(v_1, v_2, \ldots v_{29}, v_{30}|w)\mathbb{P}(w)}{\mathbb{P}(v_1, v_2, \ldots v_{29}, v_{30})}$$

- ❑ How many parameters for $P(v_1,v_2,...v_{29},v_{30}|w)$? $\sim 2^{30}$x2!
- ❑ Naive Bayes assumption: all features are independent given a class

$$\mathbb{P}(v_1, v_2, \ldots v_{29}, v_{30}|w) = \prod_{i=1}^{30} \mathbb{P}(v_i|w)$$

- ❑ Now, how many parameters for $P(v_1,v_2,...v_{29},v_{30}|w)$? 2x30 = 60
- ❑ Huge reduction of parameters, but a very strong assumption

What about when features aren't discrete? Gaussian Naive Bayes

v_1	v_2	...	v_{29}	v_{30}	w
0.22	0.80	...	0.89	0.20	"chair"
0.14	0.31	...	0.23	0.45	"celery"
0.53	0.67	...	0.01	0.43	"celery"

- Now infinite possibilities for $<v_1, v_2, \ldots v_{29}, v_{30}>$, not just 2^{30}

What about when features aren't discrete? Gaussian Naive Bayes

v_1	v_2	...	v_{29}	v_{30}	w
0.22	0.80	...	0.89	0.20	"chair"
0.14	0.31	...	0.23	0.45	"celery"
0.53	0.67	...	0.01	0.43	"celery"

- Now infinite possibilities for $<v_1,v_2,...v_{29},v_{30}>$, not just 2^{30}
- Common approach: assume $P(v_i|w)$ follows a normal (Gaussian) distribution
  - a Gaussian distribution is fully described by 2 parameters: its mean and variance

What about when features aren’t discrete? Gaussian Naive Bayes

v_1	v_2	...	v_{29}	v_{30}	w
0.22	0.80	...	0.89	0.20	“chair”
0.14	0.31	...	0.23	0.45	“celery”
0.53	0.67	...	0.01	0.43	“celery”

- Now infinite possibilities for $<v_1, v_2, ... v_{29}, v_{30}>$, not just 2^{30}
- Common approach: assume $P(v_i|w)$ follows a normal (Gaussian) distribution
    - a Gaussian distribution is fully described by 2 parameters: its mean and variance

$$\mathbb{P}(v_i = 0.22 | w = chair) = \frac{1}{\sqrt{2\pi\sigma_{i,chair}^2}} e^{\frac{1}{2}\left(\frac{0.22-\mu_{i,chair}}{\sigma_{i,chair}}\right)^2}$$

Gaussian Naive Bayes: estimation of parameters

$$\mathbb{P}(v_i = 0.22 | w = chair) = \frac{1}{\sqrt{2\pi\sigma^2_{i,chair}}} e^{\frac{1}{2}\left(\frac{0.22-\mu_{i,chair}}{\sigma_{i,chair}}\right)^2}$$

- Want to estimate $\mu_{i,chair}$, $\sigma_{i,chair}$, $\mu_{i,celery}$, $\sigma_{i,celery}$ for all i
- $\mu_{1,chair}$ = average the values of v_1 during all presentations of "chair"
- $\sigma_{1,chair}$= find standard deviation of the values of v_1 during all presentations of "chair"

Gaussian Naive Bayes: testing

- Given new values of $<v_1, v_2, \ldots v_{29}, v_{30}>$, predict what word was shown to the subject

Gaussian Naive Bayes: testing

- Given new values of <v_1,v_2,...v_{29},v_{30}>, predict what word was shown to the subject
- Using the estimated $\mu_{i,chair}$, $\sigma_{i,chair}$, $\mu_{i,celery}$, $\sigma_{i,celery}$, calculate:

$$\mathbb{P}(v_i = NEW_i | w = chair) = \frac{1}{\sqrt{2\pi\sigma^2_{i,chair}}} e^{\frac{1}{2}\left(\frac{NEW_i - \mu_{i,chair}}{\sigma_{i,chair}}\right)^2}$$

Gaussian Naive Bayes: testing

- Given new values of $<v_1,v_2,...v_{29},v_{30}>$, predict what word was shown to the subject
- Using the estimated $\mu_{i,chair}$, $\sigma_{i,chair}$, $\mu_{i,celery}$, $\sigma_{i,celery}$, calculate:

$$\mathbb{P}(v_i = NEW_i | w = chair) = \frac{1}{\sqrt{2\pi\sigma^2_{i,chair}}} e^{\frac{1}{2}\left(\frac{NEW_i - \mu_{i,chair}}{\sigma_{i,chair}}\right)^2}$$

- Then, using the Naive Bayes assumption and the Bayes theorem, calculate P (w = chair|V = NEW) and P(w =celery |V = NEW)

Gaussian Naive Bayes: testing

- Given new values of <$v_1, v_2, \ldots v_{29}, v_{30}$>, predict what word was shown to the subject
- Using the estimated $\mu_{i,chair}$, $\sigma_{i,chair}$, $\mu_{i,celery}$, $\sigma_{i,celery}$, calculate:

$$\mathbb{P}(v_i = NEW_i | w = chair) = \frac{1}{\sqrt{2\pi\sigma^2_{i,chair}}} e^{\frac{1}{2}\left(\frac{NEW_i - \mu_{i,chair}}{\sigma_{i,chair}}\right)^2}$$

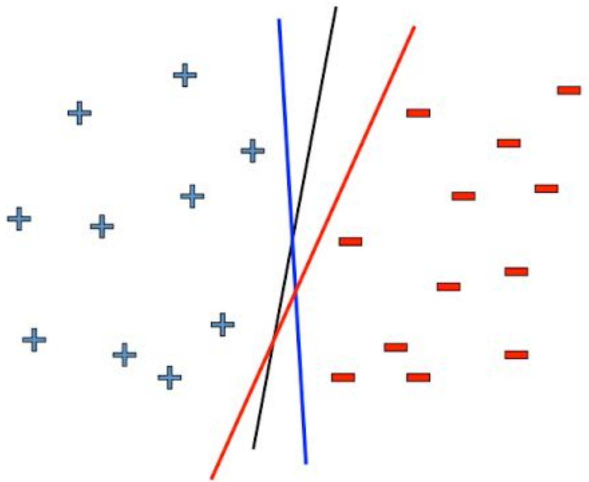
- Then, using the Naive Bayes assumption and the Bayes theorem, calculate P (w = chair|V = NEW) and P(w =celery |V = NEW)
- Assign the class with higher probability to the new data instance

Naive Bayes takeaways

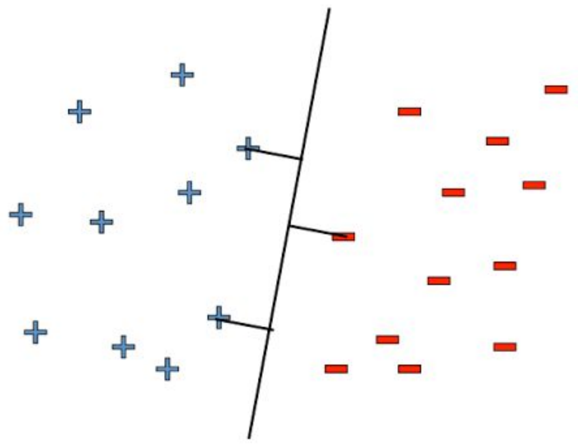
- Reduced number of parameters to estimate
- Fast training
- Can be used to quickly classify very high-dimensional data instances
- But makes strong conditional independence assumption

Support Vector Machines (SVM): motivation

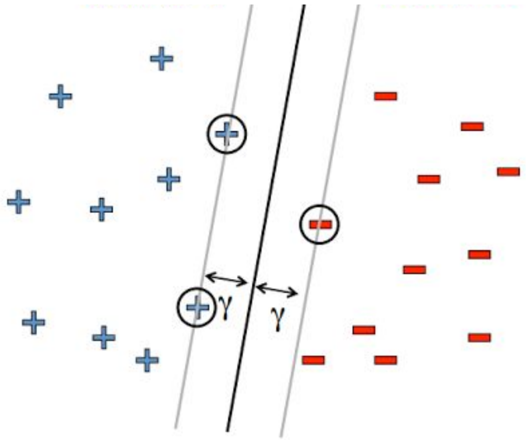
Want to directly learn a classification boundary, but which boundary is better?



SVM: choosing a decision boundary with the largest margin

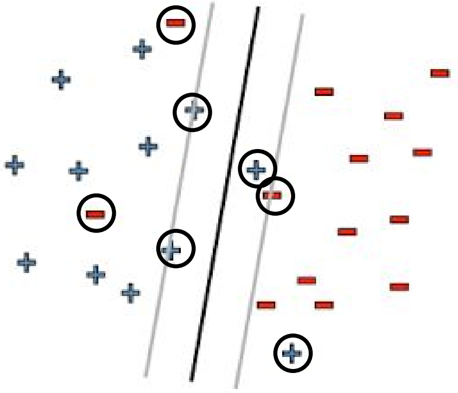


What are support vectors?



- Data points that are a margin-width away from the decision boundary
- Only need to store the support vectors to predict labels for new points => efficiency

What if data isn't linearly separable? Allow error in classification => soft-margin SVM



- Trade-off between maximizing the margin and minimizing the number of mistakes on the training data

Can we do more? Represent features in higher dimension to encourage separability

- Represent the features in higher dimension => easier to separate

Can we do more? Represent features in higher dimension to encourage separability

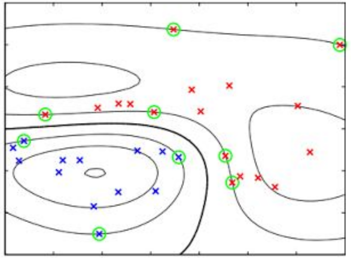
- Represent the features in higher dimension => easier to separate
- Reformulate the original SVM problem to one that depends not on the features themselves, but on the dot product of the new representations of the features

Can we do more? Represent features in higher dimension to encourage separability

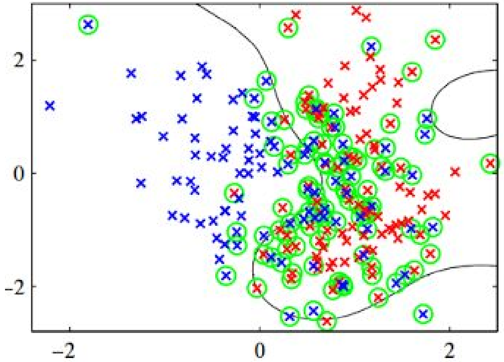
- Represent the features in higher dimension => easier to separate
- Reformulate the original SVM problem to one that depends not on the features themselves, but on the dot product of the new representations of the features
- This dot product is equal to a kernel function evaluated at the features
  - Benefit = don't have to store high-dimensional new representations of features, just need to have a way to evaluate the kernel function

Can we do more? Represent features in higher dimension to encourage separability

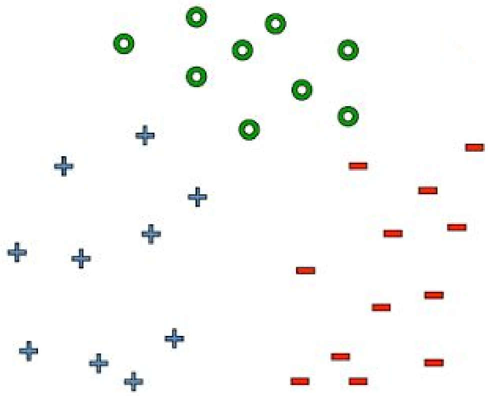
- Represent the features in higher dimension => easier to separate
- Reformulate the original SVM problem to one that depends not on the features themselves, but on the dot product of the new representations of the features
- This dot product is equal to a kernel function evaluated at the features
  - Benefit = don't have to store high-dimensional new representations of features, just need to have a way to evaluate the kernel function
  - One common kernel is the Gaussian kernel (RBF):



$$K(\mathbf{u},\mathbf{v}) = \exp\left(-\frac{||\mathbf{u}-\mathbf{v}||^2}{2\sigma^2}\right)$$



What if we have multiple classes? Multi-class SVM



- Margin = gap between correct class and nearest other class

SVM takeaways

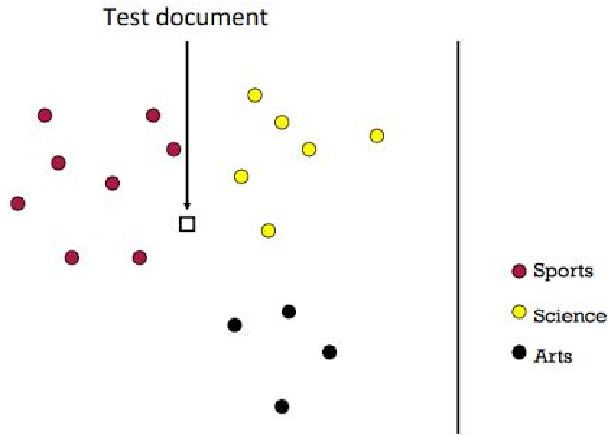
- Directly estimates the decision boundary
- Space-efficient
- Can handle non-linearly separable data through various kernel functions
- Does not directly output probabilities for classifications

k-Nearest Neighbors classifier: even fewer assumptions!

- Does not assume a model = non-parametric method
  - Number of parameters scale with the number of training data
  - Free from strong distributional assumptions that are not satisfied in practice
  - But needs lot of data to learn complex models

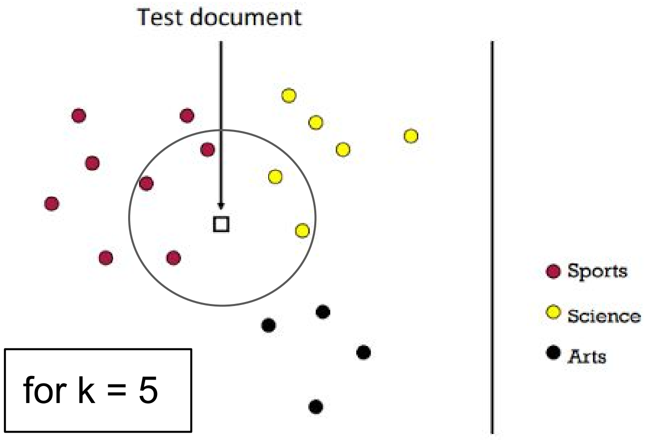
kNN: an intuitive algorithm

- We wish to classify a test instance



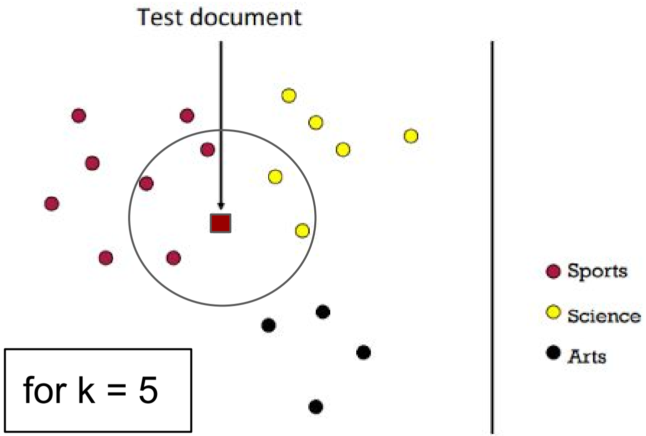
kNN: an intuitive algorithm

- We wish to classify a test instance
- Find the k closest training data instances to the test instance



kNN: an intuitive algorithm

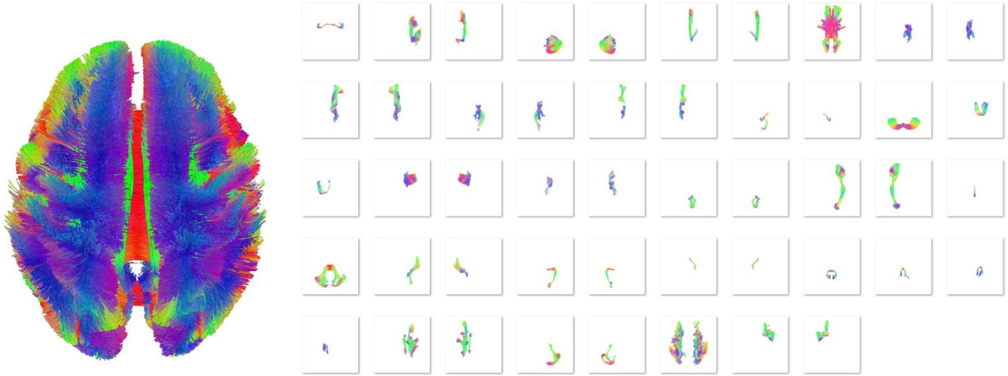
- We wish to classify a test instance
- Find the k closest training data instances to the test instance
- Assign test instance with label of the majority within the k closest instances



How to choose k?

- Trade-off between stability and accuracy
  - Larger k is more stable
  - Smaller k is more accurate
- Depends on type and amount of training data

kNN in practice: classifying DTI fibers into bundles



kNN takeaways:

- Requires a lot of data
- Requires storage and computation on entire data set
- Powerful classification technique if enough data available, and if there are no problems with data storage

Classification vs regression

- Classification = output is class label (“chair”, “celery”)
  - classify the word a subject was reading from fMRI voxels

- Regression = output is continuous value (<0.2, 0.3,...,0.9>,<0.7,0.3,...,0.1>)
  - predict next fMRI voxel values from previous voxel values

Regression: general

- Choose a parametric form for P(labels|data;θ)
- Derive a learning algorithm to estimate parameters θ

Linear regression

- Let X = data, Y = labels, and W = regression weights
- Choose linear model for P(Y|X): Y = W*X + errors

Linear regression

- Let X = data, Y = labels, and W = regression weights
- Choose linear model for P(Y|X): Y = W*X + errors
- Assume errors are normally distributed with 0 mean, and std σ
    - P(Y|X) = N(W*X , σ)

Linear regression

- Let X = data, Y = labels, and W = regression weights
- Choose linear model for P(Y|X): Y = W*X + errors
- Assume errors are normally distributed with 0 mean, and std σ
  - P(Y|X) = N(W*X , σ)
- Want to learn W from training data

$$W = \underset{W}{\operatorname{argmax}} \prod_m P(y^m|x^m, W) = \underset{W}{\operatorname{argmax}} \sum_m lnP(y^m|x^m, W)$$

Linear regression

- Let X = data, Y = labels, and W = regression weights
- Choose linear model for P(Y|X): Y = W*X + errors
- Assume errors are normally distributed with 0 mean, and std σ
  - P(Y|X) = N(W*X , σ)
- Want to learn W from training data

$$W = \underset{W}{\text{argmax}} \prod_m P(y^m|x^m, W) = \underset{W}{\text{argmax}} \sum_m lnP(y^m|x^m, W)$$

- Since: $p(y|x;W) = \frac{1}{\sqrt{2\pi\sigma^2}} \; e^{-\frac{1}{2}(\frac{y-f(x;W)}{\sigma})^2}$,

$$W = \underset{W}{\text{argmax}} \sum_m -(y^m - f(x^m;W))^2 = \underset{W}{\text{argmin}} \sum_m (y^m - f(x^m;W))^2$$

Linear regression

- Let X = data, Y = labels, and W = regression weights
- Choose linear model for P(Y|X): Y = W*X + errors
- Assume errors are normally distributed with 0 mean, and std σ
  - P(Y|X) = N(W*X , σ)
- Want to learn W from training data

$$W = \underset{W}{\operatorname{argmax}} \prod_m P(y^m | x^m, W) = \underset{W}{\operatorname{argmax}} \sum_m ln P(y^m | x^m, W)$$

- Since: $p(y|x;W) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(\frac{y - f(x;W)}{\sigma})^2}$,

$$W = \underset{W}{\operatorname{argmax}} \sum_m -(y^m - f(x^m;W))^2 = \underset{W}{\operatorname{argmin}} \sum_m (y^m - f(x^m;W))^2$$

- To solve, take derivative and set equal to 0. Closed-form solution available for simple f(x), otherwise use gradient descent

Takeaways

- 3 types of classifiers
  - Generative (e.g. Naive Bayes)
    - Make strong assumptions about data
    - Faster and need less data to estimate parameters (though, these may be wrong)

Takeaways

- 3 types of classifiers
  - Generative (e.g. Naive Bayes)
    - Make strong assumptions about data
    - Faster and need less data to estimate parameters (though, these may be wrong)
  - Discriminative (e.g. SVM)
    - Need many data instances, especially when data is high-dimensional
    - More adaptive to the actual data (few assumptions)

Takeaways

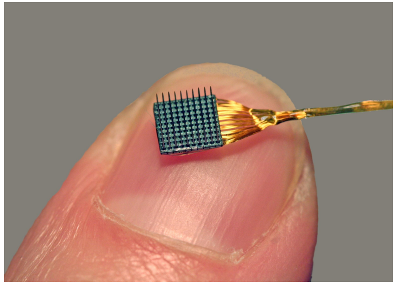
- 3 types of classifiers
  - Generative (e.g. Naive Bayes)
    - Make strong assumptions about data
    - Faster and need less data to estimate parameters (though, these may be wrong)
  - Discriminative (e.g. SVM)
    - Need many data instances, especially when data is high-dimensional
    - More adaptive to the actual data (few assumptions)
  - Instance based classifiers (e.g. kNN)
    - Require storage and computation on entire data set
    - Non-parametric so very adaptive to data, and powerful when decision boundary is irregular

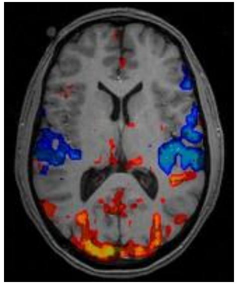
Takeaways

- 3 types of classifiers
  - Generative (e.g. Naive Bayes)
    - Make strong assumptions about data
    - Faster and need less data to estimate parameters (though, these may be wrong)
  - Discriminative (e.g. SVM)
    - Need many data instances, especially when data is high-dimensional
    - More adaptive to the actual data (few assumptions)
  - Instance based classifiers (e.g. kNN)
    - Require storage and computation on entire data set
    - Non-parametric so very adaptive to data, and powerful when decision boundary is irregular
- Regression can be used to predict continuous values

Next time: model selection & dimensionality reduction

- Deal with large number of sensors/recording sites





- investigate high-dimensional representations
  - classification (what does this high-dimensional data represent?)
  - regression (how does it represent it? can we predict a different representation?)
  - **model selection (what model would best describe this high dimensional data?)**
- **uncover few underlying processes that interact in complex ways**
  - **dimensionality reduction techniques**